MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 3, 2016/2017

TDS2111 – DATA STRUCTURES AND ALGORITHMS

(All Sections/Groups)

30 MAY 2017
02:30PM – 04:30PM
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This question paper consists of 6 pages (including cover page) with 4 structured questions.

2. Answer ALL questions. The distribution of the marks for each question is given.

3. Please write all your answers in the answer booklet provided.

Question 1

(a) Write a recursive function `getSum()` that adds the values stored in an array `x` of integers. It returns the sum of integers in `x`, stored between indexes `lower` and `upper`.

(2 marks)

Answer:

```
#include <iostream>
using namespace std;

int getSum(const int x[], int lower, int upper)
{
// TODO: Fill in your code here

}

int main(void)
{
  int x[5] = { 16, 2, 12, 40, 20 };
  cout << "The total of array is " << getSum(x,0,4) << endl;
  return 0;
}
```

Sample Output:

```
The total of array is 90.
```

Continued.....

(b) An array based `Stack` class declaration is shown in the following code. Write the code to complete the `isEmpty()` and `push()` functions. Assume that exception handling works and no additional code needs to be added. The declaration of `Stack` is given.

(4 marks)

```
const int MAX_STACK = 5;
typedef desired-type-of-stack-item StackItemType;

class Stack{

public:
   Stack();

   bool isEmpty() const;
   bool push(StackItemType newItem);
   bool pop();
   bool pop(StackItemType& stackTop);
   bool getTop(StackItemType& stackTop) const;

private:
   // array of stack items
   StackItemType items[MAX_STACK];
   // index to top of stack
   int top;
};

Stack::Stack(): top(-1){}

bool Stack::isEmpty() const{

   // TODO: Fill in your code here
}

bool Stack::push(StackItemType newItem){

   // TODO: Fill in your code here

   // if stack has no more room for another item
   if (top >= MAX_STACK-1)

   else{


      return true;
   }
}
```

Continued.....

(c) An array based Queue class declaration is shown in the following code. Write the code to complete the `isEmpty()` and `enqueue()` functions. Assume that exception handling works and no additional code needs to be added. The declaration of Queue is given.

(4 marks)

```
typedef char QueueItemType;

class Queue
{
    public:
        Queue();
        ~Queue();
        bool          isEmpty();
        void            enqueue(QueueItemType newItem);
        void            dequeue();
        QueueItemType   getFront();
    private:
        static const int MAX_QUEUE = 5;
        QueueItemType items[MAX_QUEUE];
        int front;  // index of the front element
        int back;   // index of the back element
        int count;  // number of elements in the queue
};
Queue::Queue() : front(0), back(MAX_QUEUE - 1), count(0)
{}
Queue::~Queue()
{}
bool Queue::isEmpty()
{
   // TODO: Fill in your code here
}

void Queue ::enqueue(QueueItemType newItem)
{

    if(count == MAX_QUEUE)
        cout << "queue full on enqueue";
    else
    { // queue is not full; insert item
      // TODO: Fill in your code here

    } // end if
} // end enqueue
```

Continued,.....

Question 2

Given the following list with items 18, 31, 44 and the hash function $h_1(key) = key \bmod 13$. Using an array of size 13, solve collisions using

(a) Linear probing (4 marks)

(b) Quadratic probing (2.5 marks)

(c) Double hashing with the secondary hash function $h_2(key) = 11 - (key \bmod 11)$ (3.5 marks)

Question 3

(a) Use diagrams to illustrate the linked-list implementations of the given C++ codes. (5 marks)

C++	Diagram

`struct ListNode`
`{`
`  int data;`
`  ListNode *next;`
`};`	

`int main(){`
`  ListNode *headPtr;`
`  headPtr = new ListNode;`
`  headPtr->data = 3;`
`  headPtr->next = NULL;`	

`  ListNode *temp = new ListNode;`
`  temp->data = 5;`
`  temp->next = headPtr;`
`  headPtr = temp;`	

`  ListNode *n = new ListNode;`
`  n->data = 3;`
`  n->next = temp->next;`
`  temp->next = n;`
 	
`  return 0;`
`}`	

(b) Beginning with an empty **binary search tree**, create a binary search tree when you insert the following values using the order given. [*Hint: use alphabetical order*] (5 marks)

K, G, C, H, B, Q, Y, M, D, P

Continued.....

Question 4

(a) Consider the following coding. Complete the function of `BaseEmployee`. This function should be checked with this condition: if `salary` is more than zero, keep a record of `Salary` in a new variable. Otherwise, handle the error properly. Use appropriate C++ coding to present the solution. (2 marks)

```
#include <iostream>
#include <stdexcept>
void BaseEmployee (double Salary)
{
// TODO: Fill in your code here
}
```

(b) There are plenty of applications by using shortest path algorithm. Provide at least TWO related applications. (2 marks)

(c) Georgia plans to take all subjects without repeating any of it. Below is the information about the subjects to be taken.

Table 1: Courses

Subjects Label	Pre-requisite
A	None
B	A
C	B, E
D	G
E	B, D
F	E
G	None

(i) Construct a directed graph based on Table 1. (3 marks)

(ii) A simple way to solve and find out the order of courses taken is through topological sort. Write out the algorithm of topological sort. (2 marks)

(iii) Produce the order of the courses to be taken based on the topological order you have written in (ii). (1 mark)

End of Page